SCA-Pitaya: A Practical and Affordable Side-Channel Attack Setup for Power Leakage–Based Evaluations

FRANÇOIS DURVAUX, Thales Belgium SA
MARC DURVAUX, N-labs SPRL

Although side-channel attacks appeared more than two decades ago, they remain very little discussed by security professionals outside the academia or very specific sectors (e.g., smartcard industry, governments). However, with the increasing generalisation of Internet of Things systems, they are a threat that can no longer be ignored by the operational world. This work aims to demonstrate that side-channel attacks can be practically achieved by an attacker, with reasonable means, effort, knowledge, and time. For this purpose, the contribution of this work is twofold. First, it is shown how a side-channel attack setup exploiting power leakages through electro-magnetic radiations, and making use of general-purpose and affordable equipment, can be built. The acquisition of attack power traces is made thanks to a Red Pitaya STEMLab platform coupled with a home-built radio front-end. Second, it is shown how an attack can be conducted against targets that are representative of Internet of Things devices: 8-bit and 32-bit Arduino boards.

CCS Concepts: • **Security and privacy** → **Embedded systems security**; **Side-channel analysis and countermeasures**;

Additional Key Words and Phrases: Side-channel attacks, differential power analysis, electro-magnetic radiations, Internet of Things, signal processing, heuristics

ACM Reference format:

François Durvaux and Marc Durvaux. 2020. SCA-Pitaya: A Practical and Affordable Side-Channel Attack Setup for Power Leakage–Based Evaluations. *Digit. Threat.: Res. Pract.* 1, 1, Article 3 (March 2020), 16 pages.
<https://doi.org/10.1145/3371393>

1 INTRODUCTION

Since the term *Internet of Things* (IoT) first appeared in the late 1990s, its definition has evolved, it has grown in popularity, and it has been heavily used between mainstream media and academic papers. Beyond the trendy aspects of the IoT terminology, it essentially designates infrastructures of connected objects. They are, for instance, smartphones (and more extensively all kind of smart-*something* devices), wearables, sensor nodes (e.g., industrial processes or environmental control), autonomous cars, or even medical support devices (e.g., pacemakers or insulin pumps). As a general observation, although they make our daily life easier and more comfortable, we tend to increasingly trust them with our personal data, up to our lives themselves.

From a security perspective, the IoT is truly game changing. On one hand, due to their embedded nature, IoT devices are exposed to attacks that were previously confined to very specific sectors (e.g., the smartcard industry), typically, the assumption stating that an adversary having no physical access is not valid. On the other hand, due

Authors' addresses: F. Durvaux, Thales Belgium SA, Rue des Frères Taymans 28, B-1480 Tubize, Belgium; email: francois.durvaux@be.thalesgroup.com; M. Durvaux, Rue Bois Trelong 2, B-6110, Belgium; email: marc.durvaux@gmail.com.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

© 2020 Copyright held by the owner/author(s).

2576-5337/2020/03-ART3

<https://doi.org/10.1145/3371393>

to their connected nature, compromising IoT devices may threaten complete systems, possibly leading to grave consequences. In a nutshell, the IoT enables a whole bunch of security vulnerabilities that were considered not applicable, not realistic, or too academic in the past. Today, these vulnerabilities need to be taken into account by the operational world.

Among the attacks enabled by the IoT, side-channel attacks are a serious threat because they are challenging to thwart in a cost-effective way. As a general concept, side-channel attacks take advantage of unexpected leakages of information. In other words, anything that is not standard input and output, and that can reflect what is happening “within,” can be considered as a side channel. They were first introduced with an attack against cryptographic algorithms exploiting timing [18], then power [19], electro-magnetic (EM) radiations [1, 12], or sound [14]. Yet cryptography was not always the primary target. For instance, researchers proposed sound-based key logging [3], code disassembling using the power [11], PIN inference exploiting the tilt of a phone [27], or timing attacks against cache memories [35] (recently used in Spectre and Meltdown attacks [17, 20]). These attacks are threatening whenever the security relies on a secret embedded within the device (e.g., typically a secret key).

In this article, we discuss side-channel attacks exploiting power leakages. In other words, the instantaneous power consumption of a device is exploited to retrieve sensitive information. Power-based side-channel attacks typically target (although they are not limited to) cryptographic implementations and the cryptographic keys they manipulate. Although these attacks have been studied in their very details for almost 20 years in the academic literature, they remain quite under the radar in the operational world; when security professionals have heard about them, they usually believe that they rely on unrealistic assumptions, are impractical, or are too expensive to implement. The goal of this work is to demonstrate that side-channel attacks can be practically achieved with limited cost, effort, and knowledge on targets that are representative of IoT nodes. The contribution is twofold. First, we show how a complete side-channel test bench can be built with affordable and mainstream components. For this purpose, we take advantage of classic use cases: an Advanced Encryption Standard (AES) implementation running on (i) an 8-bit ATmega328P embedded on an Arduino Uno and (ii) a 32-bit ATSAM3X8E (ARM Cortex-M3) embedded on an Arduino Due. In this particular case, we consider the 256-bit key version of the AES, which is less discussed in side-channel literature. The power leakages are measured with EM radiations emitted by the chip when it runs the algorithm. Measurements are taken with a home-made EM probe coupled with a Red Pitaya STEMLab platform (for sampling). Second, we show how the 256-bit key can be recovered within seconds using a couple of heuristics and statistics. We also emphasize the impact of using appropriate signal processing methods to recover an exploitable signal even in noisy environments.

The next sections are organized as follows. Section 2 provides background on the useful topics addressed in the rest of the article. Section 3 describes the side-channel attack setup with design trade-offs to be made. Section 4 presents how simple statistical tools can be used to recover the full 256-bit key in different adversarial contexts. Section 6 concludes the article by providing a short wrap-up of our work, as well as perspectives for future works.

2 BACKGROUND

This background section aims at providing the necessary basis for understanding the mechanisms behind side-channel attacks based on power leakages.¹ First, the structure of the AES (i.e., the cryptographic algorithm used across the article for illustration) is briefly presented. Next, insights on the source of power leakages as an exploitable side-channel are given. It is then shown how a differential power analysis (DPA) can be used to recover sensitive information from a leaking device. A popular instantiation of the attack (i.e., correlation power analysis (CPA) with a Hamming weight model) is presented. The previous works on low-cost side-channel setups eventually are discussed.

¹More advanced topics such as countermeasures, or high-order and multivariate analysis, are outside the scope of this work.

Fig. 1. Overview of the AES-256 structure.

2.1 Advanced Encryption Standard

The AES [7, 25] came out in 2001 as the reference algorithm for symmetric encryption.² The AES belongs to the block cipher family³ and processes blocks of 128 bits. Three versions of the algorithm exist, each differing by its key size (namely 128, 192, and 256 bits).

As depicted in Figure 1, the AES is made out of four main operations, each providing specific characteristics. *AddRoundKey* is the key addition layer—that is, where a round key (derived from the master key) is mixed with the input data. This operation is performed with a bit-wise XOR. *SubBytes* is a non-linear layer processing 8-bit values. *ShiftRows* and *MixColumns* both contribute to the diffusion layer, which is designed to mix all of the bits of a block. Together, these four operations form a *round*. The encryption of a 128-bit data block (i.e., a *plaintext*) is done by performing a given number of round iterations,⁴ followed by a final *AddRoundKey*. The output is referred to as the *ciphertext*. The number of iterations varies with the key size (namely 10, 12, and 14 iterations).

In the following, we focus on the AES-256 version (i.e., 14-round iterations). It is important to note that in this version, the first 128 bits of the master key are used as the first round key, and the last 128 bits are used as the second round key. In other words, the 256 bits of the master key have been processed at least once within the two first round executions.

Although side-channel attacks do not only affect cryptographic implementations, and certainly not only the AES, the latter is usually used as the typical use case in the side-channel community.

2.2 Power Leakages

Side-channel attacks exploiting power leakages are based on the general observation that the instantaneous power consumption of an electronic device depends on (i) the operation being executed and (ii) the data being processed. It is denoted as $P_{instant}$ and can be generalized as follows [9, 22]:

$$P_{instant} = P_{operation} + P_{data} + P_{algo_noise} + P_{elec_noise}.$$

²Symmetric cryptography refers to algorithms that make use of the same key for both encryption and decryption operations.

³Block ciphers designate primitives that operate on fixed-size portions of data (hence the “block”).

⁴The *MixColumns* operation is skipped in the last round.

Fig. 2. Instantaneous power consumption of an AES-128 execution [9].

Fig. 3. Data-dependent power consumption of an AES-128 execution [9].

There are four main contributions:

- $P_{operation}$ is an image of the current drawn by logic cells to perform an arithmetic operation.
- P_{data} is an image of the current caused by the internal manipulation of the data (e.g., instruction execution, memory read/write, or transition on buses).
- P_{algo_noise} refers to the algorithmic noise caused by a parasitic activity (e.g., irrelevant data processed in parallel, hardware peripherals, or multi-core CPUs).
- P_{elec_noise} refers to the electronic noise that is a combination of the physical noise (e.g., power supply, clock, radiations) and the measurement noise (e.g., quantization error).

Both $P_{operation}$ and P_{data} are potential sources of exploitable leakages. In the following, a *trace* refers to a measurement of the instantaneous power consumption. In the context of cryptographic implementations, simple power analysis (SPA) refers to attacks that exploit leakages within a single trace (possibly averaged) [19]. These attacks are particularly effective if, for example, a sequence of operations depends on sensitive data. By contrast, DPA generally refers to attacks that exploit the data dependency (i.e., P_{data}) across multiple traces [19].

Block ciphers, such as the AES, are generally safe against SPA attacks: they rely on executing the same sequence of operations, regardless of the value of the key. However, SPA is often a first interesting step for extracting relevant features of the algorithm, such as the position of the rounds. For instance, Figure 2 illustrates the operation dependency (i.e., $P_{operation}$) of an AES-128 execution. Each round can easily be identified and delimited with the repeating patterns: the 10 first peaks correspond to the rounds and the last one to the final AddRoundKey. The data dependency (i.e., P_{data}) across measured traces is illustrated in Figure 3: different data values (here 0 and 47) result in different levels of P_{data} at “leaking” time samples. Attacks exploiting differences between traces (i.e. DPA attacks) are discussed in the rest of the article.

In practice, the most common approaches for collecting power traces of a chip are either (i) measuring the voltage variations on a passive element (e.g., a resistor) connected in series with the power supply or (ii) measuring the EM radiations emitted by the power supply pin. Both are images of the varying current drawn by the target. This article focuses on the EM-based solution.

2.3 Differential Power Analysis

The AES (and all block ciphers in general) is designed such that small size blocks (typically 8 bits) can be processed independently, providing efficient implementations on all kinds of platforms. Figure 4 illustrates a typical 8-bit key addition (i.e., AddRoundKey) and S-box layer (i.e., SubBytes) in a round of the AES. For instance, in the case of the first round, the input x corresponds to a plaintext byte, and the value s corresponds to a byte of the first round key (hence a byte of the master key). The output of the AddRoundKey and the SubBytes are respectively denoted as y and z . Given that the input x is assumed to be known (as well as the AES operations), by capturing power leakages while y and z are produced, the adversary indirectly recovers information about the key byte s .

Fig. 4. AES AddRoundKey and SubBytes operations on a byte of data.

At this stage, it is important to note that the target algorithm is reasonably assumed to be known by the adversary. This assumption is based on Kerckoffs' principle [39], which states that a cryptosystem must be secure even if everything, but the key, is public knowledge. In addition, previous works have shown that the exploitation of operation dependencies (as described in Section 2.2) allows reverse engineering the sequence of operations, hence the algorithm [11].

A DPA attack follows a *divide-and-conquer* strategy. In other words, key bytes are separately attacked, then reassembled to recover the 128-bit round key.⁵ The attack on an individual key byte s is performed with the following steps [4, 23]:

- (1) The adversary captures a series of power leakage traces (corresponding to the target round) and their respective input x . As a reminder, the input x is known but not necessarily chosen. The key byte s keeps the same value for all of the traces.
- (2) For now, s is unknown, but since it is a byte, it can be easily brute forced (s is a value between 0 and 255). Therefore, for each possible value of s , the adversary predicts the intermediate values y and z .
- (3) For each prediction of z (or y), the adversary make use of a *leakage model*. The role of a leakage model is to emulate the way intermediate values are reflected in leakage traces. It can be trained (similarly to a machine learning approach) or based on *a priori* knowledge.
- (4) The modeled predictions (corresponding to a key candidate) are then compared to the observed leakages using a *distinguisher*. The correct key candidate is the one returning the highest similarity.

The efficiency of a DPA attack resides in appropriately choosing the leakage model and the distinguisher. They both depend on various parameters that are thoroughly documented in the side-channel literature. In short, they are usually a matter of compromise between the accuracy, speed, and computational requirements of an attack. It is important to note that a DPA attack can be indifferently performed on the last rounds of encryption or on the decryption operation. The only requirements for a DPA attack are (i) to know input or output bytes (plaintext or ciphertext) and (ii) having traces reflecting the operations (e.g., the rounds) where they are manipulated.

2.4 CPA with a Hamming Weight Model

CPA [5] with a *Hamming weight* model [24] is among the most popular approaches to perform a DPA attack. The Hamming weight is used as a leakage model. It requires no training, and it is based on the assumption that power leakages vary linearly with the number of 1's in the binary representation of the target value. For instance, for an 8-bit AES S-box output z , the Hamming weight is expressed as follows:

$$HW(z) = \sum_{i=0}^7 b_i \quad \text{with } z = (b_0 b_1 \cdots b_7)_2.$$

This model is usually "good enough" when investigating software implementations on microcontrollers: it generally trades accuracy for the advantage of not estimating more complex and expensive models. Regarding the distinguisher, the particularity of the CPA is that it makes use of *Pearson's correlation*. Pearson's correlation

⁵Attacking individual key bytes is a typical case. Larger pieces of the key (e.g., 16 bits) could be targeted at a higher computational cost.

between two variables X and Y is expressed as follows:

$$r = \rho(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \quad \text{with } r \in [-1; 1],$$

where cov denotes the covariance between X and Y , and σ denotes their respective standard deviation. The correlation r tends to extreme values (i.e., -1 or 1) when there is a linear relationship (i.e., a similarity) between X and Y , and tends to 0 otherwise.

Let us assume that the adversary has a vector of n observations t_i (with $i \rightarrow 1 \dots n$) corresponding a leaking time sample of the AES execution (e.g., the S-box output in the first round). It is also assumed that the adversary possesses the input bytes x_i of the leaking operation and that the key byte s (to be found) is constant. Following the steps described in Section 2.3, the adversary makes a guess s^* on the key value and then predicts the intermediate values y_i^* and z_i^* respectively corresponding to the output of the AddRoundKey and SubBytes operations. As represented in Equation (1), the adversary applies the leakage model on each predicted z_i^* and obtains the leakage predictions $HW(z_i^*)$.

$$\begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \xrightarrow{s^*} \begin{pmatrix} y_1^* \\ y_2^* \\ \dots \\ y_n^* \end{pmatrix} \xrightarrow{\text{S-box}} \begin{pmatrix} z_1^* \\ z_2^* \\ \dots \\ z_n^* \end{pmatrix} \xrightarrow{HW} \begin{pmatrix} HW(z_1^*) \\ HW(z_2^*) \\ \dots \\ HW(z_n^*) \end{pmatrix} \quad (1)$$

The adversary computes Equation (2): the correlation $r(s^*)$ between the vector of observations t_i and the predicted leakages $HW(z_i^*)$ for a key guess s^* .

$$r(s^*) = \rho \begin{pmatrix} t_1 \\ t_2 \\ \dots \\ t_n \end{pmatrix}, \begin{pmatrix} HW(z_1^*) \\ HW(z_2^*) \\ \dots \\ HW(z_n^*) \end{pmatrix} \quad (2)$$

The 256 possible values of s^* are tested following the same procedure. The best key guess s_b is the one returning the highest absolute correlation, as follows:

$$s_b = \underset{s^*}{\operatorname{argmax}}(|r(s^*)|).$$

Note that we considered a single time sample t_i , hence assuming that the adversary knows where to find the information in the leakage traces. In practice, it is common to start over the attack for each time location⁶ (e.g., within the first round).

It is also important to note that the success of the attack depends on various parameters, typically the level of noise and the number of traces. The efficiency of attacks, for a given experimentation setup, can be compared with simple tools such as the *success rate* curves⁷ [33]. The success rate returns the probability to find the right key with respect to the number of traces. The less traces an attack needs to get a high success rate, the more efficient it is.

2.5 Other Work

As a general statement, building a side-channel attack setup is essentially a matter of trade-offs between the cost, the accessibility, the performance, and the portability, among others. During the past few years, researchers have put an effort in proposing side-channel setups that would make the analysis more affordable and publicly accessible.

⁶The result of this approach with correct key guesses is further illustrated later in Figure 11.

⁷Typical success rate curves are further illustrated later in Figures 9 and 10.

The most famous example is the ChipWhisperer, which is a complete side-channel toolchain for experimenting with side-channel attacks [26]. The board has been designed in such a way that it fully integrates both the target and the measurement part. The main advantage of this platform is that it allows the user to directly focus on executing the attack. On the downside, the ChipWhisperer makes use of target-specific boards with dedicated measurement points for power leakages.

Researchers have also proposed side-channel setups that make use of consumer-grade devices, often designed initially for Digital Video Broadcasting–Terrestrial (DVB-T) receivers, as software-defined radio (SDR) modules for the trace acquisition [13, 30]. These offer the advantage of being very cheap (e.g., about 20 € for an RTL-SDR USB dongle). Given their initial target application, such devices can be tuned between typically 22 and 2,200 MHz. At the intermediate frequency (IF), the signal is sampled on 8 bits at a typically useful 2.4 MS/s [31], which is a signal bandwidth lower than 1 MHz. As these devices are USB-2 dongles, the computer interface is another limitation to the achievable maximum raw data rate.

Another family of receivers features a higher speed (and sometimes higher resolution) analog-to-digital converter (ADC) connected to an FPGA and local buffer memory. These solutions are generally more expensive (from a few hundred to thousands of €), but the increasing availability of high performance ADC tends to reduce the cost.

The setup presented in this article is based on a *Red Pitaya STEMLab 125-14* platform [29]. This device, although intended to be a general-purpose instrument, can be ranked among the second family of receivers, in the lower cost range, with a price starting from about 320 €. In its basic configuration, it provides a 125-MS/s 14-bit ADC that allows direct sampling of the signal of interest without frequency down-conversion (at least for a wide range of use cases).

Although the previous work has successfully demonstrated their attack capability, the Red Pitaya and similar acquisition platforms⁸ offer several advantages:

- baseband acquisition (direct sampling without frequency down-conversion), where most of the leaked signal energy is expected to be for low-power devices such as those encountered in IoT applications;
- wideband acquisition, which reduces the *a priori* assumptions to make on the spectral distribution of the exploitable signal;
- higher dynamic range, which reduces the need for gain adjustment in the setup that may be required to get a sufficient signal to sampling noise ratio while avoiding saturation; and
- less expensive, being much cheaper than standard oscilloscopes with similar performance.

Given the flexibility offered by this acquisition device, the probe antenna and its companion pre-amplifier can be simple and cheap as illustrated in Section 3.2.

3 EXPERIMENTAL SETUP

This section provides an extensive description of the experimental setup presented in this article. First, the target used for demonstration purposes is discussed. Second, we explain how to build a cheap probe/pre-amplifier front-end to measure power leakages through EM radiations. Finally, the acquisition of traces is described with an emphasis on the importance of the trace synchronization.

3.1 A Typical Target

To support the attack setup described in this article, an implementation of the AES-256 is used. The algorithm, described in Section 2.1, is a classic case study for demonstrating DPA attacks against cryptography (Section 2.3).

⁸ Among the multiple low-cost instrumentation platforms available on the market, independent reviewers have pinpointed two other devices with price/performance characteristics similar to the Red Pitaya. These are *Digilent Analog Discovery 2* [8] and the *Pico Technology PicoScope 2000 Series* [34]. Although the latter only features an 8-bit vertical resolution, the PicoScope 2000 series proposes versions with higher sampling frequencies (up to 1 GS/s) and larger buffers (up to 128 MS) for a slightly higher price.

The algorithm is implemented in the *C programming language*, following a “naive” 8-bit architecture, with no specific optimization.⁹

To illustrate that side-channel attacks can be indifferently applied to a wide range of microcontroller architectures, two use cases are considered. In a first use case, the AES implementation is executed by the 8-bit microcontroller *Atmel ATmega328P* embedded on the very popular *Arduino Uno* board [2]. The microcontroller chip runs at a 16-MHz clock frequency. As a second use case, the AES implementation is executed by the 32-bit microcontroller *Atmel ATSAM3X8E* (based on the high-performance *ARM Cortex-M3*) embedded on the *Arduino Due* board [2]. This second microcontroller chip runs at an 84-MHz clock frequency. It is important to note that in both cases, only the AES is executed by the microcontroller, in a completely deterministic way. In other words, no interruption nor any parallel process influencing the AES execution is involved (i.e., no operating system). In the *Arduino Due* case, the AES tables are declared with the *volatile* type to prevent leakages due to cache accesses [28]. *Arduino* projects are compiled and loaded using *Arduino IDE 1.8.9* software.

As a general behavior, the boards encrypt or decrypt data received from a host computer. The communication is done using the USB interface. Whenever the microcontroller starts executing the AES algorithm, a digital I/O is set to “high” for triggering purposes (discussed in Section 3.3).

Regarding the *Arduino Uno* target, even though one could argue that 8-bit and even 16-bit microcontrollers are losing ground due to advances in 32-bit architectures, they are still very representative of many IoT applications. In general, they are the favorite choice for cost-sensitive, very low power applications (e.g., running of dry cells) where there is no need for a lot of processing power nor a real-time operating system.

3.2 The RF Front-End

The RF front-end is made out of a probe antenna coupled with a pre-amplifier. EM probing has been preferred because it avoids intrusive modification of the target hardware and provides intrinsic insulation from the DC component. The one used in this work (home built) has the following features. The probe antenna is an air coil of 20 turns and has an inner diameter of 6.35 mm. The diameter of the loop is dimensioned with respect to the *ATmega328P* (*Arduino Uno*) such that it is close to the pin pitch. It allows selectively picking the variations of the current flowing through this microcontroller power supply pin. Note that the same antenna probe is further used for the *ATSAM3X8E* (*Arduino Due*), which has very different packaging. Of course, it is less than a theoretical optimum, but it appears that it can collect enough energy in the receiver bandwidth, as further shown in Section 4. The measured inductance at 1 MHz is 620 nH. It is connected to a differential amplifier to reject the common mode noise. Using three general-purpose bipolar transistors, the home-built pre-amplifier features a differential gain of 30 dB in a 100-kHz to 25-MHz band and a common mode gain of 0 dB. When coupled together, with the antenna in contact to a source wire, the overall gain is about 3 V/A at 1 MHz. It is important to note that this measure is very sensitive to the position of the antenna relative to the source wire. Furthermore, due to induction law, the gain increases with the frequency, as shown in Figure 5. Extensive details on the probe/pre-amplifier design are available on the GitHub page of the project [32].

3.3 Measuring Power Leakages with EM Radiations

The main instrument for the trace acquisition is the *Red Pitaya STEMLab 124-14* board used in an oscilloscope configuration [29]. The platform has been configured to sample at the rate of 125 MS/s, with the 14-bit resolution. The input range has been set to 2 Vpp. Given a data buffer size of 16,384 samples, a sample trace duration is about 131 μ s. In the case of the *Arduino Uno*, it corresponds to about 2,000 CPU clock cycles or instruction cycles (the CPU achieves near 1 MIPS/MHz). This duration is enough to span the processing time of the first two rounds of the AES algorithm. In the case of the *Arduino Due*, it corresponds to about 11,000 CPU clock cycles that cover the complete AES execution.

⁹All of the codes are available on the GitHub page of the project [32].

Fig. 5. Transfer function of the RF front-end.

In both use cases, the acquisition trigger is provided by a target I/O pin state change (i.e., low to high), as described in Section 3.1. It is important to note that synchronizing the data traces simplifies the signal processing and the attack. Since leakages of information are exploited across several traces, the trace synchronization is a strong requirement. The better traces are synchronized, the more efficient is the attack [37]. Alternatively to triggering on the target I/O, other sources of synchronization can be used, such as the start or end event of data transmission to the target. This type of event can be sensed on the transmission medium (air or wire) or at the transmitter side. In all of those alternative configurations, there is no sensing of the synchronization event at the target (i.e., the target does not need to “cooperate”).

To acquire leakage traces, the probe antenna is placed so that the coil is parallel to a power supply pin and in close contact. In this position, the supply current is not filtered by the decoupling capacitor. Requests (i.e., plaintexts) are then sent to the target to be encrypted, one by one. Every time the Red Pitaya platform has triggered, a measured trace can be recovered. The communication with the Red Pitaya (for configuration and data recovery) is done thanks to the SCPI interface.¹⁰ The Red Pitaya allows acquiring traces at a rate slightly higher than 1 trace/s.

For the following, two sets of measurements can be distinguished. The first set is identified as the *attack traces*. They are measurements for which the 256-bit key is fixed and unknown, stored in the target memory. Plaintexts are randomly chosen. Each trace corresponds to the encryption of a plaintext using the key stored within the target. The second set is identified as the *training traces*. They are measurements corresponding to the encryption of random plaintexts with random keys. Both are known and stored along with the corresponding traces. In practice, training traces can be obtained under the assumption that the adversary has prior access to a similar device (other than the target) that he or she can manipulate.

4 SIDE-CHANNEL ANALYSES

This section presents the results of CPA attacks against the two targets. Exploited traces are obtained with the side-channel attack setup described in Section 3. Different cases are considered for the attacks with varying pre-processing techniques. Practical results are presented in the following sections.

¹⁰Standard Commands for Programmable Instrumentation.

Fig. 6. Average demodulated power trace of the Arduino Uno target.

4.1 Pre-Processing of the Traces

Besides the choice of distinguisher and leakage model, applying pre-processing methods to the leakage traces may greatly impact the efficiency of DPA attacks. In general, they aim at extracting useful information from the traces while minimizing the noise. To illustrate how simple attacks such as the CPA can be devastating against targets such as the Arduino, two pre-processing methods are considered: an *amplitude demodulation* and a *projection pursuit*.

Amplitude demodulation. This method is based on the observation that EM radiations at the clock frequency and its harmonics are produced by the commutating logic. The varying number of gates switched at a particular clock cycle produces an amplitude modulation of the leaked power [1] (i.e., $P_{instant}$ as defined in Section 2.2). An asynchronous amplitude demodulation (i.e., envelope detection) is simply performed by taking the absolute value of the signal samples, followed by a low-pass filter. An amplitude demodulation is applied to traces from the Arduino Uno. The cutoff frequency of the filter is empirically set to 10 MHz. As illustrated in Figure 6, once the traces are demodulated (and averaged), the round operations can clearly be identified with the human eye.

Regarding the Arduino Due, an amplitude demodulation is not applicable. The Red Pitaya captures signals at a maximum frequency of $125/2 = 62.5\text{MHz}$, which is below the clock frequency (i.e., 84 MHz). Moreover, as displayed in Figure 7, most of the energy is located in the lower frequencies that are independent of the clock signal. Figure 8 illustrates the average trace of the two first AES round executions on the Arduino Due target.

Projection pursuit. As a second pre-processing method, this work implements a linear projection of the useful time samples using a projection pursuit approach [10]. It is based on the observation that target bytes do not leak at a single time location, as further illustrated later in Figure 11. The projection weights are heuristically chosen to optimize an objective function. In this case, the optimization aims at maximizing the absolute value of the correlation with the Hamming weights. This optimization is individually performed for both Arduino targets, thanks to 5,000 training traces (as defined in Section 3.3) with known plaintexts and known keys. In this specific case, it is assumed that the adversary can train on similar devices with a control on the key, prior to the attack. A new projection is individually computed for every byte of the key: due to the nature of AES operations, every byte has different *points of interest* in leakages traces (i.e., informative time locations).

Fig. 7. Power spectral density of the Arduino Due power traces.

Fig. 8. Average power trace of the Arduino Due target.

4.2 Recovery of a Key Byte

To illustrate the efficient of simple yet efficient pre-processing methods (described in Section 4.1), different attack cases are considered. They correspond to a CPA attack with a Hamming weight model on a set of attack traces in the following settings:

Arduino Uno:

- (1) no pre-processing
- (2) amplitude demodulation
- (3) amplitude demodulation and projection pursuit

Arduino Due:

- (1) no pre-processing
- (2) projection pursuit

Fig. 9. Success rates of CPA attacks against the Arduino Uno target (key byte 0).

Fig. 10. Success rates of CPA attacks against the Arduino Due target (key byte 0).

The efficiency of these attacks against the first key byte are summarized with the success rate curves displayed in Figures 9 and 10. The impact of the pre-processing methods is noticeable. In the case of the Arduino Uno, although a CPA on the traces with no pre-processing requires approximately 350 traces to recover a key byte with a strong certainty, it quickly drops to 200 and then 50 when demodulation and projection techniques are used. In the case of the Arduino Due, it requires approximately 70 traces to recover a key byte with the same level of certainty if no pre-processing is used and 50 traces with projection pursuit.

It is interesting to note that attacking the Arduino Due requires less traces than for the Arduino Uno, although one could expect the opposite. The origin of this difference is outside the scope of the article. However, it perfectly illustrates that the apparent complexity of a target (i.e., higher clock frequency, larger bus, deeper pipeline) does not necessarily increase the complexity of an attack.

Note that the efficiency of these attacks can be further improved with more advanced leakage models and distinguishers that are available in the side-channel literature.

Fig. 11. Correlations for key bytes 6 and 23 (first and second round) on demodulated traces (Arduino Uno).

4.3 Full Key Recovery

To perform the attack against the full AES-256 key, key bytes of a round are attacked independently. The first round is attacked in a first phase to (i) recover the first half of the master key (i.e., the 16 first bytes) and (ii) produce the input of the second round. The second round is attacked in a second phase to recover the second half of the master key (i.e., the 16 last bytes) based on the output of the first round. It is important to note that the same set of attack traces is used to attack all key bytes. As an illustration, Figure 11 displays the correlation levels output by a CPA when targeting key byte 6 (processed during the first round) and key byte 23 (processed during the second round) on the same set of traces.

Moreover, the trace acquisition and the attack phases can be done separately. In other words, once the traces are taken, the key recovery can be performed later on. In the particular setting used for these experiments, traces are measured faster than 1 trace/s (Section 3.3). Therefore, if demodulation and/or projection techniques are used, it takes less than a minute to measure enough attack traces to recover the full key. Regarding the CPA attack, the attack complexity depends on (i) the number of key byte enumerations (i.e., $32 \times 2^8 = 8,192$), (ii) the number of attack traces, and (iii) the number of samples.

For instance, recovering 32 bytes from attack traces of the Arduino Uno target, in the cases described earlier, with a regular laptop, requires the following:

- traces with no pre-processing (400 traces, 16,384 time samples): about 7 seconds;
- demodulated traces (250 traces, 16,384 time samples): about 4.3 seconds;
- demodulated and projected traces (100 traces, 1 time sample): about 0.3 second.

Attacking traces from the Arduino Due, in the same setting, takes the following:

- traces with no pre-processing (80 traces, 2,500 time samples): about 0.6 second;
- projected traces (50 traces, 1 time sample): about 0.2 second.

In cases where an attack would not allow to correctly recover all key bytes, a lack of exploitable information (with respect to the number of traces, level of noise, and the leakage model) may be compensated with key enumeration techniques [38]. This approach generally consists of testing full keys made by assembling key bytes by decreasing order of likelihood (e.g., decreasing similarity). It is important to note that applying this technique against the AES-256 is more challenging than against the AES-128 version due to the dependency between the

targeted rounds. Depending on the information returned by the side-channel analysis, the correct 256-bit key may be recovered in a reasonable time (possibly instantaneous) at a higher computational cost.

5 A NOTE ON COUNTERMEASURES AND MITIGATIONS

It is important to mention that side-channel attacks can be mitigated in different ways. Two families of countermeasures exist. The first is referred to as *hiding* and generally consists of degrading the side-channel signal-to-noise ratio to “hide” the internal activity of the chip [16, 36]. Common examples are the addition of noise in amplitude (e.g., parallel processes, peripherals) or in timing (e.g., operating system/random interrupts, dummy operations, random permutation of operations). Although hiding countermeasures are usually easy to implement with a limited impact on performance, they can be circumvented by the adversary by typically increasing the number of traces and/or with a more powerful pre-processing. The second family of countermeasures is referred to as *masking* [6, 15]. In this case, intermediate values are “mixed” with random data to decorrelate them from the power leakages. Masking techniques are very effective because they bring an exponential complexity for the adversary to succeed. However, they also come at a (exponentially) higher computation cost, they require good-quality random values, and effective implementations are not trivial (badly implemented masking schemes may lead to a false sense of security). Besides countermeasures, secure key management guidelines (e.g., no master key in end devices or frequently refreshing the keys) help in mitigating the side-channel threat.

6 CONCLUSION

As a short summary, this work shows how a side-channel attack setup, exploiting power leakages through EM radiations, can be built with an affordable and general-purpose instrumentation tool: the Red Pitaya STEMLab platform. It is also shown that, thanks to the flexibility offered by this acquisition device, the RF front-end (i.e., probe antenna and pre-amplifier) can be simple and cheap. The demonstration is made by attacking an implementation of the AES-256 running on 8-bit and 32-bit microcontrollers. By using state-of-the-art tools, the full 256-bit key can be recovered in less than a minute (acquisition and attack).

These results illustrate that the generalization of IoT systems, together with cheap instrumentation platforms, opens doors to new sort of threats and attackers. The Red Pitaya (and similar platforms) increases the threat for the defender, as much as this type of hardware is increasing in availability and performance.

About the targets, attacking microcontrollers with different architectures (namely 8 bit and 32 bit) illustrates that all platforms are affected (although with different levels of complexity). One should keep in mind that even if the adversary requires hundreds of thousands of traces, it can be achieved in days compared to brute forcing a 256-bit key. In addition, the proposed setup can be easily used on a wide variety of targets with a minimal tuning (antenna size and amplifier gain, if required) provided that there is enough leaked energy in the 0- to 50-MHz band.

About the attack, it is worth reminding that the CPA with a Hamming weight model is among the most simple approaches to perform a DPA attack and does not require *a priori* training (apart from pre-processing methods, e.g., the projection pursuit). However, it remains practically effective against commercial products. For instance, researchers have shown how a CPA attack with a Hamming weight model can be conducted against SIM cards [21]. This perfectly illustrates how an adversary can retrieve secrets from an embedded device using accessible tools.

Finally, side-channel attacks can be mitigated thanks to dedicated countermeasures and other techniques, although they usually come at a price (e.g., degraded performances) and may be tricky to implement. In any case, security evaluators should keep in mind that such threats exist and evaluate the level of risk for the exposed assets.

Future work would investigate the effort and necessary equipment to assess more advanced contexts and architectures. It would be interesting to show how non-experts can practically implement and evaluate side-channel

countermeasures. Finally, since side-channel attacks remain little known to the operational world, investigating commercial products with these analyses would help to continue raising awareness.

Note: All of the resources and codes used for this article are available on the GitHub page of the project [32].

REFERENCES

- [1] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. 2002. The EM side-channel(s). In *Cryptographic Hardware and Embedded Systems—CHES 2002*. Lecture Notes in Computer Science, Vol. 2523. Springer, 29–45. DOI : https://doi.org/10.1007/3-540-36400-5_4
- [2] Arduino. 2019. Home Page. Retrieved February 1, 2020 from <https://www.arduino.cc/>.
- [3] Dmitri Asonov and Rakesh Agrawal. 2004. Keyboard acoustic emanations. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy (S&P'04)* IEEE, Los Alamitos, CA, 3–11. DOI : <https://doi.org/10.1109/SECPR.2004.1301311>
- [4] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. 2011. Mutual information analysis: A comprehensive study. *Journal of Cryptology* 24, 2 (2011), 269–291. DOI : <https://doi.org/10.1007/s00145-010-9084-8>
- [5] Eric Brier, Christophe Clavier, and Francis Olivier. 2004. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems—CHES 2004*. Lecture Notes in Computer Science, Vol. 3156. Springer, 16–29. DOI : https://doi.org/10.1007/978-3-540-28632-5_2
- [6] Jean-Sébastien Coron and Louis Goubin. 2000. On Boolean and arithmetic masking against differential power analysis. In *Cryptographic Hardware and Embedded Systems—CHES 2000*. Lecture Notes in Computer Science, Vol. 1965. Springer, 231–237. DOI : https://doi.org/10.1007/3-540-44499-8_18
- [7] Joan Daemen and Vincent Rijmen. 2000. Rijndael for AES. In *Proceedings of the AES Candidate Conference*. 343–348.
- [8] Analog Discovery 2 (Digilent). 2019. Home Page. Retrieved February 1, 2020 from <https://www.analogdiscovery.com/>.
- [9] François Durvaux. 2015. *Towards Fair Side-Channel Security Evaluations*. Ph.D. Dissertation. Université Catholique de Louvain.
- [10] François Durvaux, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Jean-Baptiste Mairy, and Yves Deville. 2015. Efficient selection of time samples for higher-order DPA with projection pursuits. In *Constructive Side-Channel Analysis and Secure Design*. Lecture Notes in Computer Science, Vol. 9064. Springer, 34–50. DOI : https://doi.org/10.1007/978-3-319-21476-4_3
- [11] Thomas Eisenbarth, Christof Paar, and Björn Weghenkel. 2010. Building a side channel based disassembler. *Transactions on Computational Science* 10 (2010), 78–99. DOI : https://doi.org/10.1007/978-3-642-17499-5_4
- [12] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. 2001. Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded Systems—CHES 2001*. Lecture Notes in Computer Science, Vol. 2162. Springer, 251–261. DOI : https://doi.org/10.1007/3-540-44709-1_21
- [13] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. 2015. Stealing keys from PCs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In *Cryptographic Hardware and Embedded Systems—CHES 2015*. Lecture Notes in Computer Science, Vol. 9293. Springer, 207–228. DOI : https://doi.org/10.1007/978-3-662-48324-4_11
- [14] Daniel Genkin, Adi Shamir, and Eran Tromer. 2014. RSA key extraction via low-bandwidth acoustic cryptanalysis. In *Advances in Cryptology—CRYPTO 2014*. Lecture Notes in Computer Science, Vol. 8616. Springer, 444–461. DOI : https://doi.org/10.1007/978-3-662-44371-2_25
- [15] Jovan D. Golic and Christophe Tymen. 2002. Multiplicative masking and power analysis of AES. In *Cryptographic Hardware and Embedded Systems—CHES 2002*. Lecture Notes in Computer Science, Vol. 2523. Springer, 198–212. DOI : https://doi.org/10.1007/3-540-36400-5_16
- [16] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. 2006. An AES smart card implementation resistant to power analysis attacks. In *Applied Cryptography and Network Security*. Lecture Notes in Computer Science, Vol. 3989. Springer, 239–252. DOI : https://doi.org/10.1007/11767480_16
- [17] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. 2018. Spectre attacks: Exploiting speculative execution. *meltdownattack.com*. Retrieved February 1, 2020 from <https://spectreattack.com/spectre.pdf>.
- [18] Paul C. Kocher. 1996. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology—CRYPTO 1996*. Lecture Notes in Computer Science, Vol. 1109. Springer, 104–113. DOI : https://doi.org/10.1007/3-540-68697-5_9
- [19] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *Advances in Cryptology—CRYPTO 1999*. Lecture Notes in Computer Science, Vol. 1666. Springer, 388–397. DOI : https://doi.org/10.1007/3-540-48405-1_25
- [20] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, et al. 2018. Meltdown: Reading kernel memory from user space. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security'18)*. 973–990. <https://www.usenix.org/conference/usenixsecurity18/presentation/lipp>.
- [21] Junrong Liu, Yu Yu, François-Xavier Standaert, Zheng Guo, Dawu Gu, Wei Sun, Yijie Ge, and Xinjun Xie. 2015. Small tweaks do not help: Differential power analysis of MILENAGE implementations in 3G/4G USIM cards. In *Computer Security—ESORICS 2015*. Lecture Notes in Computer Science, Vol. 9326. Springer, 468–480. DOI : https://doi.org/10.1007/978-3-319-24174-6_24

- [22] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2007. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer.
- [23] Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. 2011. One for all—All for one: Unifying standard differential power analysis attacks. *IET Information Security* 5, 2 (2011), 100–110. DOI : <https://doi.org/10.1049/iet-ifs.2010.0096>
- [24] Thomas S. Messerges and Ezzy A. Dabbish. 1999. Investigations of power analysis attacks on smartcards. In *Proceedings of the 1st Workshop on Smartcard Technology (Smartcard'99)*. <https://www.usenix.org/conference/usenix-workshop-smartcard-technology/investigations-power-analysis-attacks-smartcards>.
- [25] National Institute of Standards and Technology (NIST). 2001. *Advanced Encryption Standard. FIPS PUB 197*. NIST.
- [26] Colin O'Flynn and Zhizhang (David) Chen. 2014. ChipWhisperer: An open-source platform for hardware embedded security research. In *Constructive Side-Channel Analysis and Secure Design*. Lecture Notes in Computer Science, Vol. 8622. Springer, 243–260. DOI : https://doi.org/10.1007/978-3-319-10175-0_17
- [27] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. 2012. ACCessory: Password inference using accelerometers on smartphones. In *Proceedings of the 2012 Workshop on Mobile Computing Systems and Applications (HotMobile'12)*. ACM, New York, NY, 9. DOI : <https://doi.org/10.1145/2162081.2162095>
- [28] Dan Page. 2002. Theoretical use of cache memory as a cryptanalytic side-channel. *Cryptology ePrint Archive: Report 2002/169*. Retrieved February 1, 2020 from <http://eprint.iacr.org/2002/169>.
- [29] Red Pitaya. 2019. Home Page. Retrieved February 1, 2020 from <https://www.redpitaya.com/>.
- [30] Pieter Robyns. 2019. Performing Low-Cost Electromagnetic Side-Channel Attacks Using RTL-SDR and Neural Networks. Retrieved February 1, 2020 from <https://www.rtl-sdr.com/sdr-talks-from-fosdem-2019-em-attacks-with-rtl-sdr-and-neural-networks-sdr-astronomy-telescopes-gnu-radio-in-2019-and-more/>.
- [31] RTL-SDR. 2019. Home Page. Retrieved February 1, 2020 from <https://www.rtl-sdr.com/about-rtl-sdr/>.
- [32] GitHub. 2019. SCA-RedPitaya. Retrieved February 1, 2020 from <https://github.com/fdurvaux/sca-redpitaya>.
- [33] François-Xavier Standaert, Tal Malkin, and Moti Yung. 2009. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology—EUROCRYPT 2009*. Lecture Notes in Computer Science, Vol. 5479. Springer, 443–461. DOI : https://doi.org/10.1007/978-3-642-01001-9_26
- [34] Pico Technology. 2019. PicoScope 2000 Series. Retrieved February 1, 2020 from <https://www.picotech.com/oscilloscope/2000/picoscope-2000-overview/>.
- [35] Yukiyasu Tsunoo, Teruo Saito, Tomoyasu Suzaki, Maki Shigeri, and Hiroshi Miyauchi. 2003. Cryptanalysis of DES implemented on computers with cache. In *Cryptographic Hardware and Embedded Systems—CHES 2003*. Lecture Notes in Computer Science, Vol. 2779. Springer, 62–76. 62–76.
- [36] Michael Tunstall and Olivier Benoît. 2007. Efficient use of random delays in embedded software. In *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*. Lecture Notes in Computer Science, Vol. 4462. Springer, 27–38. DOI : https://doi.org/10.1007/978-3-540-72354-7_3
- [37] Jasper G. J. van Woudenberg, Marc F. Witteman, and Bram Bakker. 2011. Improving differential power analysis by elastic alignment. In *Topics in Cryptology—CT-RSA 2011*. Lecture Notes in Computer Science, Vol. 6558. Springer, 104–119. DOI : https://doi.org/10.1007/978-3-642-19074-2_8
- [38] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. 2012. An optimal key enumeration algorithm and its application to side-channel attacks. In *Selected Areas in Cryptography*. Lecture Notes in Computer Science, Vol. 7707. Springer, 390–406. DOI : https://doi.org/10.1007/978-3-642-35999-6_25
- [39] Auguste Kerckhoffs. 1883. La cryptographie militaire [Military cryptography]. *Journal Des Sciences Militaires* IX (1883), 5–38.

Received May 2019; revised September 2019; accepted November 2019